Hanwen Zhang
CS526 Data Structures and Algorithms
Term Project - Simulation Program

Discussion:

This project is a small simulation to demonstrate the implementation of the priority scheduling algorithm using priority queue. The program reads from an input file and processes data from the file to store its priority as the key and the process object as value with corresponding id, arrivalTime, duration, and schedules the order at which they are executed based on their priorities. During execution, only one process can run at a time, causing a wait time for the other processes. Time is measured in loop iterations of the program (one loop of the program = one time unit). One process executing at a time, and the higher priority process is the main focus which will be run first. To work around the issue like higher priority takes too long to run, the program increases the priority of the lower priority by one if their queue waiting timer reaches the maximum waiting time. In this case, the program implements a max waiting time of 30 iterations.When running the ProcessScheduling Class, an output text file will be generated in the src folder, and all the processes will be printed onto this text file.

The data structure I used in project is a pre-defined HeapAdaptablePriorityQueue given by the course, which already included the use of a Comparator interface to order each process based on the priority, which higher priority runs first. With this type of data structure, we can work around the issue like if one process takes too long to run that solely based on priority causes a bottleneck, the program could increases the priority of the lower priority by one if their queue waiting timer reaches the maximum waiting time. In this case, the program implements a max waiting time of 30 iterations.

For processes that had equal priority, it may have been better to execute the process with earlier arrival time instead of choosing arbitrarily. At a high level, I would modify my project to accommodate this by having another comparator to compare the arrival time when the processes have equal priority to make sure that the one with earlier arrival time can be finished first in the program to avoid any bottleneck or long waits.

I would consider making changes to my project to improve efficiency or readability or reusability by splitting big classes into smaller classes, such as the file *ProcessPriorityQueue.java* where the input file is read, process objects are created and stored into a priority queue, and the process activity is written to an output file, I would split these features each to a class for future uses to improve reusability, also instead of read file and write file with a prefilled file and file name, I would make a parameter to pass in the function where users can read any file and write to any file as needed. I would also improve the code written in *ProcessScheduling.java* - the main driver where simulation runs to improve its efficiency and readability.